UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/911,819	07/24/2001	John T. Micco	04899-046001	6291

7590 10/19/2007

Kevin J. Canning, Esq.
Lahive & Cockfield, LL.P
28 State Street
Boston, MA 02109

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
10/19/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/911,819	MICCO ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 August 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-3,5-32 and 34-56* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,5-32 and 34-56* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 8/10/07.

As indicated in Applicant's response, claims 1, 11, 17, 21, 29, 39, 45, 49 have been amended, and claims 4, 33 canceled. Claims 1-3, 5-32, 34-56 are pending in the office action.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

3. Claims 1-3, 5-7, 11-13, 17, 20, 29-32, 34-35, 39, 40-41, 45, and 48 are rejected under 35 U.S.C. 102(b) as being anticipated by Shannon et al., "Mapping the Interface Description Language Type Model in C", November 1989, IEEE Transactions on Software Engineering, Vol. 15, No. 11 (hereinafter Shannon).

As per claim 1, Shannon discloses in a computing device, a method comprising:

providing a definition of a function associated with a first language (e.g. ...*data structures found in procedural programming languages* – 2nd para, R col, pg. 1334; *function ... parameters* – middle R, pg. 1334 – Note: creating IDL constructs for a function is a ABC compiler reads on providing definition of function in one language prior to using it the IDL to translate a corresponding C function - *Ada ...to C , Ada Breadboard Compiler* - pg. 1334, R col.);

creating description information (e.g. IDL – Introduction, pg. 1333, L col.) about the function from the definition of a function associated with a first language (e.g. *existing*

languages ... LISP, Diana structures ... mapped Ada ...to C, Ada Breadboard Compiler – Introduction pg. 1333-1334; *...data structures found in procedural programming languages –* 2nd para, R col, pg. 1334; *node declaration ... function => name; String; parameters: Seq of ... formal parameters*, pg. 1334, right column; *class, function* - Fig. 1 pg. 1335; *Process declaration* - Fig. 2);

storing the description information in a storage device (e.g. IDL specifications – Fig. 3, pg. 1336), where the description information allows the call to the function in the first language to be associated with a call (e.g. *procedure calls ... returned value* – pg 1340, L col. bottom) to a corresponding function in a second language (*Ada ...to C , Ada Breadboard Compiler* - pg. 1334, R col.; *C compiler ...function Afunction, Vfunction*, pg. 1340);

identifying a call to the function in the first language (e.g. *statement –> Function, function Afunction* - pg. 1339, top R); retrieving the stored description information for the function from the storage device (e.g. Fig. 3, pg. 1336); and

translating the call to the function in the first language into the call to corresponding function in thee second language using the description information, wherein translating uses the description information (e.g. *procedure calls ... returned value* – pg 1340, L col. bottom; *Ada ...to C , Ada Breadboard Compiler* - pg. 1334, R col; *C compiler ...function Afunction, Vfunction*, pg. 1340) instead of the definition of the function.

As per claim 2, Shannon discloses a file of description items and derived description information (e.g. sections II, III – pg. 1334-1339; IDL specifications - Fig. 3).

As per claim 3, Shannon discloses

examining the definition of the function associated with the first language (e.g. ...*data structures found in procedural programming languages* – 2nd para, right column, pg. 1334; *node declaration ... function => name; String; parameters: Seq of ... formal parameters*, pg. 1334, right column – Note: the fact of creating appropriate parameters for a function disclose examining definition of a function in a source language);

and derive information (*IDL specifications* - Fig. 3, pg. 1336; *IDL specification ... specifically, C macros ... and functions* – pg. 1334, left column, 4th para – Note: generating of IDL from existing functions written in some languages inherently teaches derive information for said functions) about the function.

As per claim 5, Shannon discloses creation of C language constructs, hence has implicitly disclose the *.lib* files associated with the assembling of object files prior to linking in C. Therefore, Shannon has implicitly disclosed library wherein entries associated with the assembling process in C compiler because at the time the invention was made it was a known concept that C compiler create lib files during a pre-linking compiler process.

As per claims 6 and 7, Shannon discloses a declaration with a set of input and output port (ch. B – pg. 1335; Fig. 2, pg. 1336) and input/output mapping (pg. 1338, Private types - right column); hence has disclosed analysis of the first language so to derive input/output formal parameters leading to creation of C formal parameters, i.e. list of input or output/return parameters otherwise the target function declaration would not result in a correct function declaration (Note: the declaration of formal input and output, and scope of variables declared in a function in 4th generation like C was a known concept at the time the invention was made; and

according to C code translation by Shannon, this reads on input/outputs as in a formal declaration of a function as set forth in claims 1, 3).

As per claim 11, Shannon discloses in a computing device, a method comprising: providing a file of description items (e.g. *...data structures found in procedural programming languages* – 2nd para, R col, pg. 1334; *function ... parameters* – middle R, pg. 1334 – Note: creating IDL constructs for a function is a ABC compiler reads on providing definition of function in one language prior to using it the IDL to translate a corresponding C function - *Ada ...to C , Ada Breadboard Compiler* - pg. 1334, R col.), where one or more items include description information about one or more functions associated with a first language (e.g. *procedure calls ... returned value* – pg 1340, L col. bottom; *Ada ...to C , Ada Breadboard Compiler* - pg. 1334, R col; *C compiler ...function Afunction, Vfunction*, pg. 1340),

identifying a call to the function in the first language; retrieving an item from the file of description items (e.g. *statement –> Function, function Afunction* - pg. 1339, top R; Fig. 3, pg. 1336);

using the description information for the function to translate a call to the function in the first language into a call to a corresponding function in a second language, wherein translating uses the description information (e.g. *procedure calls ... returned value* – pg 1340, L col. bottom; *Ada ...to C , Ada Breadboard Compiler* - pg. 1334, R col; *C compiler ...function Afunction, Vfunction*, pg. 1340) instead of the definition of the function; and using the file of description items to translate a first program file into a second program file (Fig. 3 – Note: C compiler using a ABC tool - *Ada Breadboard Compiler* - pg. 1334, R col - reads on second program file compiled from using ABC IDL specifications).

As per claims 12-13, referring to rationale of claims 6-7, Shannon has disclosed analysis of the first language input/output formal parameters leading to creation of C formal parameters, i.e. list of input or output/return parameters otherwise the target function declaration would not result in a correct function declaration (Note: the declaration of formal input and output, and scope of variables declared in a function in 4th generation like C was a known concept at the time the invention was made; and according to C code translation by Shannon, this reads on input/outputs as in a formal declaration of a function as set forth in claims 1, 3).

As per claim 17, Shannon teaches storing the translated call in the second program file (Note: a function call being translated – see *executable* - Fig. 3, pg. 1336 - signifies file being stored after compiler translation leading to an executable).

As per claim 20, refer to rationale as set forth in claims 12-13.

As per claim 29, Shannon teaches computer program product, tangibly stored on a computer-readable medium, for creating a data file, the product comprising instructions operable to cause a programmable processor to:

obtain a definition of function associated with a first language (Note: creating definition in IDL construct for a function reads on obtaining definition of a function in that language - *Ada ...to C , Ada Breadboard Compiler* - pg. 1334, R col; *C compiler ...function Afunction, Vfunction*, pg. 1340); create description information about the function from the definition of the function associated with a first language (refer to claim 1);

store the description information in a storage device, where the description information allows the call to the function in the first language to be associated with a call to a corresponding function in a second language (refer to claim 1);

identify a call to the function in the first language; retrieve the stored description information for the function from the storage device (refer to claim 1); and

translate the call to the function in the first language into a call to the corresponding function in a second language using the description information, wherein translating uses the description information (refer to claim 1) instead of the definition of the function.

As per claims 30-32, 34-35, these are the computer product claims corresponding to claims 2-3, 5-6; hence are rejected with the corresponding rejections as set forth therein respectively.

As per claim 39, this is the computer-medium product version of claim 11, hence is rejected with the corresponding rejection as set forth therein.

As per claims 40-41, these are the computer product claims corresponding to claims 12-13; hence are rejected with the corresponding rejections as set forth therein respectively.

As per claim 45, this is the computer-medium product version of claim 17, hence is rejected with the corresponding rejection as set forth therein.

As per claim 48, refer to rationale as set forth in claims 12-13.

4. Claims 8-10, 14-16, 18-19, 36-38, 42-44 and 46-47 are rejected under 35 U.S.C. 103(a) as being unpatentable over Shannon et al., "Mapping the Interface Description Language Type Model in C", November 1989, IEEE Transactions on Software Engineering, Vol. 15, No. 11, in view of Bjarne Stroustrup, "the C++ Programming Language", 2nd Edition, copyright 1991 (hereinafter Stroustrup).

As per claim 8, Namespace and function scope involving local and global parameters are known in 4th generation like C. Stroustrup discloses C programming language and namespace

for addressing scope of members of a function dictated within a declaration directives of such namespace (see pg. 634, chp. 3.3.1). Hence, it would have been obvious for one skill in the art at the time the invention was made in light of Shannon type checking and preprocessing for runtime efficiency analyze a function scope while creating of description information about a function of the first language in order to derive a correct scope when effecting a C function declaration according to Stroustrup; because this would support the scope of definition of parameters by Shannon's C directives because this helps support scope of members within a class or function definition as endeavored by Shannon's preprocessing macro directives (see Shannon: pg. 1341-1344).

As per claims 9 and 10, Shannon does not explicitly teach determining of variable arguments in a function and a variable return of results. But according to Stroustrup's teaching of C++ providing a variable number of arguments, e.g. input arglist[], or argv[]; or a variable output/return in form of an array, or pointer to a struct or linked list (Stroustrup: *argv[]*– chp. 3.1.6, pg. 87; pg. 485 chp. 3.4; *argument ... list* - chp. 8.2.5 – pg. 532), this a known concept C construction at the time the invention was made. Hence, for one skill in the art at the time the invention was made, determining whether a function to have a variable input arguments or return variables would also have been obvious according the above teaching by Stroustrup to address possibility where number of arguments can vary, and analyzing and precisely controlling on such extensibility and variability of parameters as endeavored by Shannon' s type and code checking for runtime as set forth above would allow the target code to accommodate for such eventuality, using the known approach provided by C as mentioned above.

As per claims 14-16, refer to the rationale of claims 8-10.

As per claims 18-19, these claims subject matter fall under the ambit of the subject matter of claims 15-16; and are rejected using to rationale as set forth in claims 15-16, respectively.

As per claims 36-38, these are the computer product claims corresponding to claims 14-16; hence are rejected with the corresponding rejections as set forth therein respectively.

As per claims 42-44, these are the computer product claims corresponding to claims 14-16; hence are rejected with the corresponding rejections as set forth therein respectively.

As per claims 46-47, these claims correspond to the subject matter of claims 15-16; and are rejected using to rationale as set forth in claims 15-16, respectively.

5. Claims 21-28, and 49-56 are rejected under 35 U.S.C. 103(a) as being unpatentable over Elmroth et al., "A Web Computing Environment for the SLICOT Library", December 2000, Brite-Euram III, Networks Programme NICONET, in view of Research Systems, "IDL", copyright 1994, further in view of Shannon et al., "Mapping the Interface Description Language Type Model in C", November 1989, IEEE Transactions on Software Engineering.

As per claim 21, Elmroth discloses a method in a computing device comprising:

providing a library file including functions defined by a first language (e.g. *SLICOT Library, BLAS, LAPACK* – pg. 1, Introduction; *Riccati equations* - Fig. 2-3; *...uploaded ... Matlab files, data files, Latex, Scilab* – pg. 2, pg. 6, Fig. 1, 4 – Note: uploaded matrices in Matlab or Latex format , or Riccati equations read on library files - i.e. files served as input to a library generating tool - defined in one language, all such file integrated into the SLICOT library file framework);

creating a function library (e.g. *SLICOT Routines* – Fig. 6) and a description file (e.g. *library... algorithms*, Fig. 1, pg. 2; PHP scripts, Aux Routines – pg. 6-7 – Note: library of algorithms coupled with php scripts to support an interactive a top level web page interface reads on description of user request via a session, the realization of algorithms which is implemented via low level libraries routines), the function library including one or more functions defined by a second language, each function in the function library being translated version of a function in the library file (e.g. *SLICOT routines* – pg. 6; *...written in C or Fortran* – pg.8: Conclusions and Future Work), identifying a call to the function in the first language and retrieving the description file (see Fig. 6, pg. 7 – Note :web interface with collecting of PHP content reads on retrieving of description based on request parsing and underlying calls to auxiliary libraries) and

using the description file to translate a program file from the first language into the second language (e.g. *SLICOT routines, Matlab binaries, PHP scripts* – pg. 7-8 – Note: library files xxxxMD or xxxOD files in conjunction with PHP scripts and converting math functions into m-files read on one or more functions translated from the library file in a second language, i.e. Matlab binaries function files)

But Elmroth does not explicitly disclose the description file including description information that enables translation of a call to the function in a first language into a call to a corresponding function in a second wherein translating uses the description information instead of the definition of the function; nor does Elmroth disclose that each call in the program file to a function in a first language is translated into a call to a corresponding function in the second language. Converting declarative functionality based on specification file (*library... algorithms*, Fig. 1, pg. 2) to realize the algorithms or models of math functionality or complex problems (see

Elmroth: Figure 2, pg. 3) into another program like high-level programming language like *Fortran or C* (see Elmroth – pg. 8: Conclusions and Future Work) as proposed by Elmroth was a known concept at the time the invention was made. Research_Systems, discloses intermediate descriptive language representation (called as interactive data language very similar to declarative framework file by Elmroth) and mapping various application functions to a fourth-generation programming language like C, and using such a descriptive data file to implement applications similar to the Riccati Equations or Matlab files by Elmroth, e.g. Math functions, graph plotting, Image Processing (see Research_Systems, pg. 1-5). In case Elmroth were to use the declarative framework and implement complex algorithms as programming language like C including C interface and call routines, a similar description language representation to enable such conversion would be needed based on well known practices as conveyed above; e.g. converting functions in one language into a corresponding functions in another language, like from Math functions to C program as taught by Research_Systems, or in scenario wherein C routines or procedures would have to be implemented as proposed by Elmroth: that is, a intermediate (interface language) description file as taught by Shannon as evidenced by an IDL specification (refer to claim 1). One would be motivated in doing so that the interactive or intermediate description information (as by Research_Systems) when used instead of the PHP scripts, would enable translation of a call to the function in a math algorithm into a call to a corresponding function in a second language like C as suggested by Elmroth using Shannon's IDL description file, without requiring processing of the definition of the C function; such that each call in the first language program (math functions in a declarative input domain – see Elmroth Fig. 2-3) file is translated into a call to a corresponding function in the second language;

e.g. a C routine (as taught in Shannon). The benefits of using this intermediate description file would have been obvious because of the same reasons listed by Shannon via using Shannon's IDL: provides type safe implementation into a high-level programming like C, and further of the very nature of Elmroth declarative approach, would not require processing of the definition of the function (see Shannon: Introduction, pg. 1333-1334); that is, alleviating writing C code from scratch in that it would otherwise require analyzing --by the code writer-- all the integral details pertinent to each C routine definition.

As per claim 22, this claim includes the limitation as to translate a call to the function in a first language into a call to a corresponding function in a second language as in claim 21; hence is rejected as set forth therein.

As per claim 23, this claim limitation corresponds to that the description file that would be used to obviate deriving information at framework; hence the generating of the IDL by Shannon entails that deriving information from C routine definition has been disclosed (see Claim 1). In view of the combination Elmroth and Research_Systems and Shannon from above, this *examining* and *deriving* limitations are rejected as obvious using most of Shannon teachings, mapping of function using a description file derived from examining a function in the first language library file, in light of Research_Systems endeavor to translated into C code a declarative math model or algorithms.

As per claims 24 and 25, these claims correspond to limitations of claim 3, 4 and 11; hence are rejected using the corresponding Shannon's teachings in view of the rationale to combine Elmroth, Research_Systems and Shannon as set forth above.

As per claim 26-28, Elmroth discloses a Web call mapping and converting interface (Fig. 6), while Shannon discloses an interface to check call for error and type violation (the User Interface – pg. 1344). In light of the rationale as to combine the IDL teachings by Shannon with Elmroth's web interface and PHP scripts, the motivation to provide Elmroth' s Web interface a function evaluation interface as suggested by Shannon would have been for the same reasons as combining Elmroth with Shannon as in claim 21. Further, Elmroth does not specify variable input descriptor, variable output descriptor, descriptor for a known number of input or output arguments as recited in claims 18-20. In view of the rationale to combine Elmroth with the IDL by Research_Systems and Shannon, these claims will be rejected as in claims 18-20 respectively.

As per claim 49, this is the computer-medium product version of claim 21, hence is rejected with the corresponding rejection as set forth therein.

As per claims 50-56, these are the computer product claims corresponding to claims 22-28; hence are rejected with the corresponding rejections as set forth therein respectively.

Response to Arguments

6. Applicant's arguments filed 8/10/07 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

35 USC § 102 Rejection:

(A) Applicants have submitted that Shannon does not disclose 'retrieving stored description information for the function…', and that Shannon's IDL for representing intermediate representations between phases of various compilers and for mapping purposes such that IDL specifications are not used for "translating the function in the first language into a call to the

corresponding function in the second language" as required in claim 1 (Appl. Rmrks pg. 12, bottom, pg. 13, top para). The rejection has addressed how the IDL by Shannon has been retrieved as a stored file or specifications and submitted into a C compiler framework for construct validation in compliance to the target language. The rejection has mapped the translating limitation with Shannon's portions that show function being identified in a IDL form then translated into a C form. The weight of the *call to a function* limitation has been interpreted as the very implementation mechanics of a function, wherein such implementation includes parameters needed for invoking such call; e.g. input and return type *parameters*; hence the 'call' limitation is deemed fulfilled according to the cited parts --mentioning about parameters for a function -- in Shannon set forth in the rejection. Besides, a 'call to a function' cannot only becomes an actual runtime call when the environment in which a call is effectuated is runtime; and the claim for what is worth, is describing a framework purported to translate mere code construct from one form onto another, therefore, there is no executable code being evidenced therein to lend the fact that a 'call' is a runtime call. The argument is therefore not persuasive.

(B) Applicants have submitted that Shannon's IDL is not first language; but IDL specifications being processed cannot disclose 'uses the description information instead of the definition of the function' (Appl. Rmrks pg. 13, 3rd para). The argument does not seem to be conveying a clear rationale; and applicants failed to point out how specifically the part cited in the rejection would not disclose this "instead of definition of the function" requirement. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

(C) Applicants have submitted that Shannon's IDL is only for passing structures among cooperating processes as an independent language, thus cannot disclose 'identifying a call to the function in the first language' nor can this be a first language by virtue of Shannon's Fig. 3 (Appl. Rmrks 4th para). The 'call to a function' has been addressed in section A; and the first language has been clearly explained in the Rejection as the first language (e.g. Ada) serving as basis for the (language independent) description to be constructed in terms of a IDL specifications, such that this specification is inputted into the process of Figure 3 by Shannon whereby translating into a target C construct would yield a executable file in this target language. Shannon's IDL is a description file to enable a function form defined and identified in the above first language to be converted into a second language corresponding function form; hence Shannon is deemed as fulfilling the above limitations.

(D) As per claim 11, Applicants' argument revolves about how Shannon's IDL for only containing basic structures cannot be used for translating calls. The issue about a 'call to a function' in light of the observations in section C in conjunction with Figure 3 IDL has addressed a 'call to a function' and *description file* used to translate a second language function into executable, when a first language form of function has been identified from using the IDL descriptive information (e.g. Figure 3 of Shannon). The Applicants' argument for claim 11 (Appl. Rmrks pg. 14, bottom) is therefore not convincing.

USC § 103 Rejection:

(E) Applicants have submitted that notwithstanding Shannon's not fulfilling 'translating the call to the function in the first language ... second language using the description information', Stroupstrup when combined also fails to disclose 'translating uses ... instead of the definition of

the function', thus not remedying to the above shortcomings by Shannon (Appl. Rmrks pg. 16, and pg. 17, 2nd para). The rationale of obviousness has started by identifying what is not explicit in Shannon, and in so doing, uses the C language teachings by Stroustrup to overcome the list of parameters (*) which is missing in Shannon; next the rejection has set forth how one of ordinary skill in the art would implement Shannon's C compiler to incorporate description for construct having arguments list as well-known in C programming language. The rejection is not addressing the 'call to a function' or 'uses the description ... instead of' limitations which appear to be Applicants' main focus in the above argument. A rationale of obviousness has its own substeps, and Applicants fail to identify where the Office in establishing these steps fail to fulfill the teaching about the missing element (see *) in the main reference that require a combination thereof with another reference. Applicants' argument does not seem to be compliant with a proper prima facie case of rebut with the particulars of a 103 rejection; hence is deemed insufficient to overcome how Shannon in light of Stroustrup fulfill the argument list limitation.

(F) For the arguments against claims 14-16, 18-19, 36-38, 42-44, 46-47, 21-28, 49-56, these amount to rehash of the points raised earlier and addressed in sections A-C above.

(G) Applicants have submitted that for claim 21, that Elmroth does not teach 'description information ... call to a function in the first language ... second language' and that the IDL description file by Research systems is not the same as Shannon's IDL as asserted by the Examiner (Appl. Rmrks pg. 19). The rejection has first pointed out how a intermediate description file would be used to support a declarative framework by which math functionality can be translated into programming language written in C. Next, Research Systems has been used to evidenced that another declarative framework using a description file to support a similar

endeavor as Elmroth; then, in light of the conversion suggested in Elmroth, the rationale set forth use of a description file as in Research_Systems in view of the IDL by Shannon, such that this description file can be used to support translation of math domain functions into C programming constructs, for the very reason that code developers would not spend extraneous effort in understanding C functions very definition by the C language, notably based on Shannon's approach. Applicants fail to identify where the Office in establishing these steps fail to fulfill the teaching about the missing element (description file to enable translation of function from 1st language -- or math functions-- into C language) in the main reference that require a combination thereof with another reference.

(H) Last, the arguments in terms of how the references do not teach 'description file' about function (Appl. Rmrks pg. 20); and in light of the response set forth above, the arguments are not persuasive in proving how Shannon fails to teach a file when parsed an used in a translation engine enables function otherwise written a first language can be derived and reconverted into a second language equivalent form; and that has been cited throughout the rejection. Concerning the obviousness rationale using Elmroth, Research_Systems, and Shannon, Applicants appear to denigrate the 103 rationale by merely attacking each reference separately, and in so doing, make a big issue out of the nomenclature discrepancy regarding the IDL acronym without (i) explaining how a difference underlying such acronym can impact the obvious rationale as established (refer to section G) and (ii) specifically pointing how the rationale as set forth in the Office Action particularly fail to fulfill the claimed subject matter as questioned.

The claims will stand rejected as set forth in the Office Action.

Conclusion

7. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
October 17, 2007